ELEC 204 Digital Design Lab Report
LAB-4
Name: Barış KAPLAN (ID NUMBER: 69054)
Date: 23 May 2021

The objectives of the lab-4

1-) Learning how to construct a sequential circuit which includes the control inputs and the state outputs for the purpose of changing the LED states by shifting 1 from left to right or from right to left with an amount of 1-bit place.
2-) Learning how to utilize FPGA for the purpose of the sequential circuit implementation.
3-) Learning how a sequential circuit works.
4-) Getting used to the design of the sequential circuits.
5-) Learning how to write the VHDL code of the sequential circuits.

What my code has to do

At the beginning, my code should check whether we have rising edge or not. Inside this check, my code should check the values of the direction bits and the current 10-bit LED display. If the LED display is equal to 1000000000, and the current value of the direction bit is equal to 0,the push button input of the left user is equal to 1; my code should update the state of the LED display to 0100000000, should update the value of the direction bit to 0, and should increment the score of the left user by 1. If the LED display is equal to 0100000000 and the current value of the direction bit is equal to 0, my code should update the state of the LED display to 0010000000, and should update the current value of the direction bit to 0. Until we reach the case where the 10-bit LED display is equal to 0000000001, my code should update the value of the direction bit to 0, and should shift the bit which is 1 from left to right with an amount of 1-bit place. My code also has to keep track of the direction bit. Until we reach the case where the 10-bit LED display is equal to 0000000001, my code should update the value of direction bit to 0 (While the bit '1' is going from left to right, my code assumes the value of the direction bit 0. While the bit '1' is going from right to left, my code assumes the value of the direction bit 1). When we reach the case where the 10-bit LED display is equal to 0000000001, my code should update the value of the direction bit to 1. Because the bit which is 1 will start to go from its' right-most position to its' left-most position(from 0000000001 to 1000000000). Moreover, when the push button input of the right user is equal to 1 and the state of the 10-bit LED display is equal to 0000000001, my code should increment the value of the right score by 1. After that, my code should update the value of the direction bit to 1, and should shift the bit which is 1 from right to left with an amount of 1-bit place until we reach the case where the 10-bit LED display is equal to 1000000000. If the 10-bit LED display is equal to 1000000000, since the bit which is 1 will start to go from left to right, my code should update the value of the direction bit to 0.

How I did it

At the beginning, I have defined 3 inputs and 3 outputs. Then, I have defined some intermediate signals. I have defined an intermediate signal called 'LedRegion' to represent the current 10-bit led state. The intermediate signal called 'directionBit' represents the direction. In order to enable the arithmetic operations with scoreLeft representing the left user score and scoreRight representing the right user score, I have defined the intermediate signals called 'scoreLeft' and 'scoreRight' as unsigned. I have defined a process with the parameter called 'MCLK'. At the beginning of the process, I have checked whether we have rising edge or not by using the rising_edge function with a parameter of 'MCLK'. Inside this check, I have checked the 10-bit Led states and the values of the direction bits. I have done these checks in if and elsif statements. If the led state is 1000000000, the direction bit is equal to

0, and the push button of the left user is 1; I have updated the LedRegion to 0100000000, updated the value of direction bit to 0, and incremented the signal called ‘scoreLeft’ by 1. Until the LedRegion is equal to 0000000001 and the direction bit is equal to 1; I have continued to update the direction bit 0, and update the LedRegion to the version where 1 is shifted from left to right with an amount of 1 bit place. If the LedRegion is equal to 0000000001, the direction bit is equal to 1, and the push button of the right user is equal to 1; I have updated the LedRegion to 0000000010, updated the direction bit to 1, and incremented the signal called ‘scoreRight’ by 1. Until the LedRegion is equal to 1000000000 and the direction bit is equal to 0; I have continued to update the direction bit 1, and update the LedRegion to the version where 1 is shifted from left to right with an amount of 1-place. If the LedRegion is equal to 1000000000, I have updated the direction bit to 0 in order to start the process again.

THE INPUTS

The names of the inputs are PushBt1, PushBt2, and MCLK. The input which is called ‘PushBt1’ represents the push button of the left user. The input which is called ‘PushBt1’ is a 1-bit input. Moreover, the type of the input called ‘PushBt1’ is STD_LOGIC. The input which is called ‘PushBt2’ represents the push button of the right user. The input which is called ‘PushBt2’ is a 1-bit input. Moreover, the type of the input called ‘PushBt2’ is STD_LOGIC. The input which is called ‘MCLK’ is a 1-bit input. Moreover, the type of the input called ‘MCLK’ is STD_LOGIC.

THE OUTPUTS

The names of the outputs are LedDisplay, SevenSegment1, and SevenSegment2. The LedDisplay is a 10-bit output. The SevenSegment1 is a 7-bit output. The Seven Segment2 is a 7-bit output. The type of the output called ‘LedDisplay’ is STD_LOGIC_VECTOR. The type of the output called ‘SevenSegment1’ is STD_LOGIC_VECTOR. The type of the output called ‘SevenSegment2’ is STD_LOGIC_VECTOR. The output called ‘LedDisplay’ represents the current situation of the 10-bit Led pattern. The output called ‘SevenSegment1’ represents the score of the led user. The output called ‘SevenSegment2’ represents the score of the right user.

WHAT THE VHDL CODE MUST DO

When the state of the led display is 1000000000 and the value of the push button input of the left user is 1; the vhdl code should increment the score of the left user by 1, make the value of the direction bit 0, and make the state of the led display 0100000000. When the state of the led display is 0100000000 and the value of the direction bit is 0; the vhdl code should update the value of the direction bit to 0, and make the state of the led display 0010000000. By applying the same logic, until we reach the case where the led display is equal to 0000000001; the vhdl code should check the state of the led display and the direction bit. Depending on these checks, the vhdl code should update the value of the direction bit to 0, and it should also shift the bit ‘1’ from left to right with an amount of 1 bit place. If the led display is 0000000001, the vhdl code should update the value of the direction bit to 1 because the bit 1 will start to go to the opposite direction now. When the state of the led display is 0000000001 and the value of the push button input of the right user is 1; the vhdl code should increment the score of the right user by 1, make the value of the direction bit 1, and make the state of the led display 0000000010. When the state of the led display is 0000000010 and the value of the direction bit is 1; the vhdl code should update the value of the direction bit to 1, and make the state of the led display 0000000100. By applying the same logic, until we reach the case where the led display is 1000000000; the vhdl code should check the state of the led display and the direction bit. Depending on these checks, the vhdl code update the direction bit to 1, shift the bit 1 from right to left with an amount of 1 bit place. If the led display is 1000000000, the vhdl code should update the value of the direction bit to 0 because the bit 1 will start to go to the opposite direction now.

HOW MY CODE WORKS

Initially, I have defined a process with the parameter of MCLK. By using the rising_edge function, I have checked whether we have rising edge or not. In this check, I have checked the states of the 10-bit led display and the values of the direction bit. At the start, the state of the led display is 1000000000. If the state of the led display is 1000000000 and the value of the push button of the left user is 1; I have increased the score of the left user by 1, updated the value of the direction bit to 0, and made the state of the led display 0100000000. Until I reach the situation where the bit 1 is at its' right-most position(led display=0000000001); I have updated the value of the direction bit to 0, and I have also shifted the bit which is 1 from left to right with an amount of 1 bit place. If the led display is 0000000001 and the value of the push button of the right user is 1; I have incremented the score of the right user by 1, updated the direction bit to 1, and made the state of the led display 0000000010. By using the same logic; until the state of the led display is 1000000000; I have shifted the bit 1 from right to left with an amount of 1 bit place, and updated the value of the direction bit to 1. If I reach the case where the state of the led display is 1000000000; in order to make the 10-bit led pattern start again, I have updated the direction bit to 0.

THE TRUTH TABLE

A	B	C	D	E	F	G	H	J	M	N_A	N_B	N_C	N_D	N_E	N_F	N_G	N_H	N_J	N_M	R	N_R	P1	P2
1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1(0)	0,1
0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0,1	0,1
0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0,1	0,1
0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0,1	0,1
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0,1	0,1
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0,1	0,1
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0,1	0,1
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0,1	0,1
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0,1	0,1
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	1	1	0,1	1(0)
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	1	1	0,1	0,1
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0,1	0,1
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	0,1	0,1
0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	0,1	0,1
0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0,1	0,1
0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0,1	0,1
0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0,1	0,1
0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0,1	0,1

A,B,C,D,E,F,G,H,J, and M represent the bits of the current 10-bit led state.
N_A,N_B,N_C,N_D,N_E,N_F,N_G,N_H,N_J, and N_M represent the bits of the next 10-bit led state.
R represents the direction bit.
N_R represents the next state of the direction bit.
P1 represents the push button of the left user.
P2 represents the push button of the right user.
Incr_1 represents the amount of increment of the score of left user.
Incr_2 represents the amount of increment of the score of right user.
Note: You can find the remaining part of the truth table below.

R	N_R	P1	P2	Incr_1	Incr_2
0	0	1(0)	0,1	1	0
0	0	0,1	0,1	0	0
0	0	0,1	0,1	0	0
0	0	0,1	0,1	0	0
0	0	0,1	0,1	0	0
0	0	0,1	0,1	0	0
0	0	0,1	0,1	0	0
0	0	0,1	0,1	0	0
0	1	0,1	0,1	0	0
1	1	0,1	1(0)	0	1
1	1	0,1	0,1	0	0
1	1	0,1	0,1	0	0
1	1	0,1	0,1	0	0
1	1	0,1	0,1	0	0
1	1	0,1	0,1	0	0
1	1	0,1	0,1	0	0
1	1	0,1	0,1	0	0
1	0	0,1	0,1	0	0

1. Problems encountered, errors and warnings resolved

Explain what problems you encountered while writing your code.
I have not encountered with any errors while writing my code.

Explain what synthesis errors and warnings you observed.
I have not encountered with any synthesis errors and warnings.

Explain what problems you had to solve (or could not) on your board even if your code could be synthesized successfully.
Since I did not use the FPGA board, I have not encountered with the problems that I had to solve on my FPGA board.

2. Conclusion

SUMMARY OF THE LAB-4

In the LAB-4, I have implemented a led ping pong game by changing the states of the 10-bit led display and the direction bit. If the bit 1 is going from left to right; I have updated the value of the direction bit to 0, and shifted 1 from left to right with an amount of 1 bit place. If the bit 1 is going from right to left; I have updated the direction bit to 1, and shifted 1 from right to left with an amount of 1 bit place. From left to right, I have assumed that the value of the direction bit is equal to 0. From right to left, I have assumed that the value of the direction bit is equal to 1. When I reach the case where led display is 1000000000 and the value of the push button of the left user is 1; I have incremented the score of the left user by 1 and updated the value of the direction bit to 0. When I reach the case where led display is 0000000001 and the value of the push button of the right user is 1; I have incremented the score of the right user by 1, and updated the value of the direction bit to 1. At the end of the process,

I have converted the types of the scoreLeft and scoreRight to std_logic_vector and assigned them to SevenSegment1 and SevenSegment2 respectively. Moreover, I have assigned the output called 'LedDisplay' to the intermediate signal called 'LedRegion'. To determine whether the right user or the left user wins, I have checked the values of the SevenSegment1 and SevenSegment2 from the simulation results. I have observed that we can change the winner of the game by making the time of switch debouncing different for the left user and the right user.

WHAT I HAVE LEARNED FROM LAB-4

1-) I have learned how to implement switch debouncing in VHDL.
2-) I have learned how to implement bit shifting in VHDL.
3-) I have learned how to write the test bench code for a clock synchronized process.
4-) I have learned how to use process in the VHDL code.
5-) I have learned how to use the unsigned type in VHDL.
6-) I have learned that I can do arithmetic operations on the unsigned and signed types in VHDL.
7-) I have learned how to convert an unsigned type to std_logic_vector.
8-) I have learned the construction steps of a sequential circuit.

References

1. Please cite any resource (web site, book, youtube video) you used for this lab.

Youtube video: https://www.youtube.com/watch?v=z4eqE7srNyU&list=PLZv8x7uxq5XY-IQfQFb6mC6OXzz0h8ceF&index=23
Youtube video: https://www.youtube.com/watch?v=z6Biw6xai1E
Youtube video: https://www.youtube.com/watch?v=F6P1suRDcWc
Youtube video: https://www.youtube.com/watch?v=2YbelsD79Q4

LAB-4 VHDL SOURCE CODE

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

use IEEE.NUMERIC_STD.ALL;

library UNISIM;

use UNISIM.VComponents.all;


entity Lab4Code is

   Port ( PushBt1 : in  STD_LOGIC;

          PushBt2 : in  STD_LOGIC;

          MCLK : in  STD_LOGIC;

          LedDisplay : out  STD_LOGIC_VECTOR (9 downto 0);

          SevenSegment1 : out  STD_LOGIC_VECTOR (6 downto 0);

         SevenSegment2: out STD_LOGIC_VECTOR(6 downto 0));

end Lab4Code;
```

```
architecture Behavioral of Lab4Code is

signal LedRegion : std_logic_vector(9 downto 0):="1000000000";

signal directionBit : STD_LOGIC:='0';

signal scoreLeft : unsigned(6 downto 0):="0000000";

signal scoreRight : unsigned(6 downto 0):="0000000";

begin
process(MCLK)
begin
if(rising_edge(MCLK)) then
   if(LedRegion="1000000000" and directionBit='0') then
              if(pushBt1='1') then
                     scoreLeft<=scoreLeft+1;
                     end if;
                     LedRegion<="0100000000";
                     directionBit<='0';
            end if;
            elsif(LedRegion="0100000000" and directionBit='0') then
                     LedRegion<="0010000000";
                     directionBit<='0';
            elsif(LedRegion="0010000000" and directionBit='0') then
                     LedRegion<="0001000000";
                     directionBit<='0';
             elsif(LedRegion="0001000000" and directionBit='0') then
                     LedRegion<="0000100000";
                     directionBit<='0';
             elsif(LedRegion="0000100000" and directionBit='0') then
```

```
              LedRegion<="0000010000";
              directionBit<='0';
       elsif(LedRegion="0000010000" and directionBit='0') then
              LedRegion<="0000001000";
              directionBit<='0';
       elsif(LedRegion="0000001000" and directionBit='0') then
              LedRegion<="0000000100";
              directionBit<='0';
       elsif(LedRegion="0000000100" and directionBit='0') then
              LedRegion<="0000000010";
              directionBit<='0';
       elsif(LedRegion="0000000010" and directionBit='0') then
              LedRegion<="0000000001";
                    directionBit<='1';
             elsif(LedRegion="0000000001" and directionBit='1') then
              if(pushBt2='1') then
              scoreRight<=scoreRight+1;
              end if;
              LedRegion<="0000000010";
   directionBit<='1';
             elsif(LedRegion="0000000010" and directionBit='1') then
               LedRegion<="0000000100";
                     directionBit<='1';
             elsif(LedRegion="0000000100" and directionBit='1' ) then
               LedRegion<="0000001000";
                    directionBit<='1';
             elsif(LedRegion="0000001000" and directionBit='1') then
               LedRegion<="0000010000";
                     directionBit<='1';
             elsif(LedRegion="0000010000" and directionBit='1') then
```

```
                                LedRegion<="0000100000";
                                            directionBit<='1';
                            elsif(LedRegion="0000100000"and directionBit='1') then
                                LedRegion<="0001000000";
                                            directionBit<='1';
                            elsif(LedRegion="0001000000"and directionBit='1') then
                                LedRegion<="0010000000";
                                            directionBit<='1';
                            elsif(LedRegion="0001000000" and directionBit='1') then
                                LedRegion<="0010000000";
                                            directionBit<='1';
                            elsif(LedRegion="0010000000" and directionBit='1') then
                                LedRegion<="0100000000";
                                            directionBit<='1';
                            elsif(LedRegion="0100000000" and directionBit='1') then
                                LedRegion<="1000000000";
                                          directionBit<='0';
                            end if;
end process;
 LedDisplay<=LedRegion;
 SevenSegment1<=std_logic_vector(scoreLeft);
 SevenSegment2<=std_logic_vector(scoreRight);

end Behavioral;
```

<u>LAB-4 SIMULATION CODE(TEST BENCH CODE)</u>

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

--Uncomment the following library declaration if using
```

```
-- arithmetic functions with Signed or Unsigned values
--USE ieee.numeric_std.ALL;

ENTITY Lab4Simulation IS
END Lab4Simulation;

ARCHITECTURE behavior OF Lab4Simulation IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT Lab4Code
    PORT(
         PushBt1 : IN  std_logic;
         PushBt2 : IN  std_logic;
         MCLK : IN  std_logic;
         LedDisplay : OUT  std_logic_vector(9 downto 0);
         SevenSegment1 : OUT  std_logic_vector(6 downto 0);
         SevenSegment2 : OUT  std_logic_vector(6 downto 0)
        );
    END COMPONENT;


   signal PushBt1 : std_logic := '0';
   signal PushBt2 : std_logic := '0';
   signal MCLK : std_logic := '0';

   signal LedDisplay : std_logic_vector(9 downto 0);
   signal SevenSegment1 : std_logic_vector(6 downto 0);
    signal SevenSegment2 : std_logic_vector(6 downto 0);
```

```
  constant MCLK_period : time :=10 ns;

BEGIN
   uut: Lab4Code PORT MAP (
         PushBt1 => PushBt1,
         PushBt2 => PushBt2,
         MCLK => MCLK,
         LedDisplay => LedDisplay,
         SevenSegment1 => SevenSegment1,
         SevenSegment2 => SevenSegment2
        );
  MCLK_process :process
  begin
                MCLK <= '0';
                wait for MCLK_period/2;
                MCLK <= '1';
                wait for MCLK_period/2;
  end process;
  stim_proc: process
  begin

    wait;
  end process;
           PushBt1<='0','1' after MCLK_period/2;
           PushBt2<='0','1' after MCLK_period/2;

END;
```